3. Система команд микропроцессора

3.1. Классификация команд

Под *командой* понимают совокупность сведений, необходимых процессору для выполнения определенного действия при реализации программы. Множество команд, реализуемых в ЭВМ, образует *систему команд*, выбор которой является важнейшей задачей проектирования ЭВМ. Система команд определяет область применения и эффективность микропроцессорной системы управления. Несмотря на то, что подавляющее большинство алгоритмов может быть реализовано посредством ограниченного набора команд, большинство ЭВМ имеет 60–120 базовых команд. Под базовой понимают команду, которая определяет выполняемую операцию без учета модификаций данной команды за счет использования различных режимов адресации. Например, МП КР580ВМ80А имеет 78 базовых команд, однако с учетом модификаций число команд равняется 224. Это позволяет в ряде случаев существенно сократить длину программ, а следовательно, уменьшить время решения задачи и размер программы в памяти. Таким образом, система команд определяет возможности машины.

Теоретически ограничения на число команд ЭВМ нет; например, при введении команд из нескольких слов можно выделить больше бит под код операции. Каждый дополнительный бит в коде операции удваивает число команд. С другой стороны, чем сложнее команда, тем быстрее выполняется программа из-за сокращений числа обращений к памяти.

Классификация команд по основным признакам представлена на рис. 12.

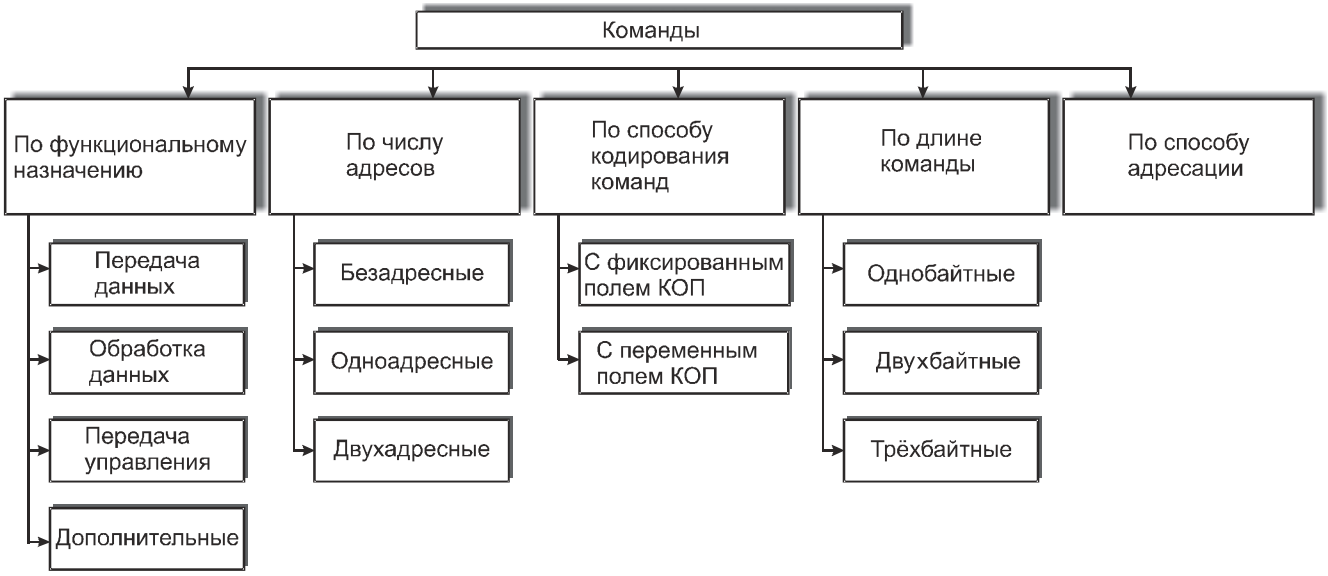

Рис. 12. Классификация команд

Систему команд рассматриваемого микропроцессора КР580ВМ80А можно классифицировать по трем основным признакам:

1. *Длине команды* (одно-, двух- и трехбайтные);

2. *Функциональному назначению* (передачи, обработки данных, команды управления);

3. *Архитектурным признакам* (операции с регистрами, памятью и портами).

Современные тенденции развития ЭВМ показывают, что фирмы-разработчики микропроцессоров стараются создавать дополнительные наборы команд на основе уже существующих, сохраняя программную преемственность с предыдущими поколениями процессоров. Такие ресурсоемкие задачи, как расчет трехмерной графики, компрессия/декомпрессия аудио-видеоданных и другие, используют дополнительные наборы команд (3DNow, MMX, SSE, и др.), оптимизированные под соответствующие приложения.

3.2. Методы адресации

Для взаимодействия различных модулей в микроЭВМ должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров МП и регистров устройств ввода/вывода. Поэтому каждой из запоминающих ячеек присваивается адрес, т.е. однозначная комбинация бит. Количество бит определяет число идентифицируемых ячеек. Обычно ЭВМ имеет различные адресные пространства памяти и регистров МП, а иногда – отдельные адресные пространства регистров, устройств ввода/вывода и внутренней памяти. Кроме того, память хранит как данные, так и команды. С другой стороны, при разработке микропроцессоров стараются использовать коды операций минимальной длины, что приводит к возникновению проблемы идентификации данных из-за короткого машинного слова. Поэтому для ЭВМ разработано множество способов обращения к памяти, называемых режимами адресации.

Режим адресации памяти – это процедура или схема преобразования адресной информации об операнде в его исполнительный адрес. В микропроцессоре КР580ВМ80А используется пять методов адресации:

1. **Прямая** – в команде задается адрес ячейки памяти, где расположен операнд; он указывается во втором (младшая часть адреса) и в третьем (старшая часть) байтах команды. К этой группе также относятся команды, в которых задается адрес порта ввода/вывода:

STA 8020H – требует четырех обращений к памяти;

IN 05H – требует двух обращений к памяти.

2. **Прямая регистровая** – в команде задается адрес регистра или пары регистров, где находится 8- или 16-битный операнд:

MOV A, B – требует одного обращения к памяти;

CMP C.

3. **Непосредственная** – операнд содержится в самой команде:

MVI A, 08H – требует двух обращений к памяти;

LXI M, 8020H – требует трех обращений к памяти.

4. **Косвенная** – адрес M ячейки памяти, где расположен операнд, определяется содержимым парного регистра, явно или неявно указанного в команде:

MOV A, M – пересылка в A из ячейки памяти, на которую указывает HL;

LDAX B – загрузка A из ячейки памяти, на которую указывает пара BC.

5. **Неявная** – адрес операнда не указывается в явном виде, а определяется кодом операции:

ADD B; A ← A+B, аккумулятор не задается в явном виде.

Следует отметить, что в одной команде могут использоваться два различных метода адресации, например, в команде MVI A, 08H используется прямая регистровая адресация для приемника и непосредственная для источника. В системах реального времени для повышения скорости вычислений программ необходимо максимально использовать *регистровую* адресацию.

3.3. Формат команд

Формат команды определяет ее структурные элементы, каждый из которых интерпретируется определенным образом при выполнении команды. Среди таких элементов (полей) выделяют:

- код операции, определяющий выполняемое действие;
- адрес ячейки памяти, регистра процессора, внешнего устройства;
- режим адресации;
- операнд при использовании непосредственной адресации;
- код анализируемых признаков для команд условного перехода.

Почти во всех форматах команд первые биты отводятся для кода операции, но далее форматы команд разных ЭВМ сильно отличаются друг от друга. Остальные биты должны определять операнды или их адреса, и поэтому они используются для комбинации режимов, адресов регистров, адресов памяти, относительных адресов и непосредственных операндов. Обычно длина команды варьируется от 1 до 3 и даже 6 байт.

Число бит, отводимое под КОП, является функцией полного набора реализуемых команд. При использовании фиксированного числа бит под КОП для кодирования всех m команд необходимо в поле КОП выделить $\log_2 m$ двоичных разрядов. Так как информация берется только из одной ячейки, эта ячейка называется источником; ячейка, содержимое которой изменяется, называется приемником.

Средняя длина команды в типичной программе для 8-разрядного микропроцессора равна двум байтам, а для программ более поздних 16-разрядных процессоров типа i8086 она равна 4,1. Поэтому на программах с большим

количеством логических операций 8-разрядные процессоры незначительно уступают 16-разрядным.

Положение полей в микропроцессоре КР580ВМ80А переменное, и в зависимости от команды, назначение поля может иметь следующее значение:

Byte 1 – содержит код операции, длину команды, адреса регистров (рис. 13);

Byte 2 – содержит адрес порта ввода/ вывода, 8-разрядный операнд или младшую часть 16-разрядного операнда;

Byte 3 – содержит старшую часть 16-разрядного операнда.

Рис. 13. Формат первого байта команды КР580ВМ80А

Многобайтовая команда должна размещаться в последовательно расположенных ячейках памяти.

Адрес приемника (DDD) и адрес источника (SSS) задают регистр или ссылку на ячейку памяти (признак косвенной адресации), при этом адреса регистров кодируются следующим образом (табл. 1):

Таблица 1

Кодировка адресов регистров

Регистр R	**Код SSS или DDD**	**Регистр R**	**Код SSS или DDD**	**Регистровая пара RP**	**Код RP**
B	000	H	100	BC	00
C	001	L	101	DE	01
D	010	M	110	HL	10
E	011	A	111	SP	11

3.4. Команды пересылок

Команды пересылки выполняют передачу данных из регистра в регистр и размещение данных в памяти. Они не формируют флаги, выполняются за один машинный цикл и кодируются одним байтом по следующим правилам:

1) сначала указывается приемник, затем источник (в архитектуре фирмы DEC – наоборот);

2) если источник или приемник – это ячейка памяти, то ее адрес размещается в регистровой паре HL.

MOV R1, R2 – пересылка (копирование) содержимого регистра R2 в регистр R1. Режим адресации – регистровый. Команда выполняется за один машинный цикл и имеет формат

01’DDD’SSS,

где 01 – код операции пересылки, DDD – номер регистра назначения, а SSS – номер регистра приемника. Например, команда MOV A, B; A ← B, примет следующий двоичный вид: 01’111’000, что соответствует шестнадцатеричному коду 78H.

MOV R, M – пересылка содержимого ячейки памяти, адрес которой находится в регистровой паре HL в регистр R. Режим адресации – косвенный регистровый. Так как требуется обращение к памяти, то команда выполняется за два машинных цикла. Система команд данного микропроцессора не имеет команд для передачи данных между ячейками памяти.

MVI – пересылка (MoVe Immediately) однобайтного непосредственного операнда, содержащегося во втором байте команды в регистр или память. Режим адресации – непосредственный. Команда состоит из двух байт, выполняется за два машинных цикла и имеет формат

00’DDD’110.

Например, команда MVI B,05H; B ← 05H представляет собой два последовательно расположенных в памяти байта – код операции (06H) и (05H). Команда примет следующий двоичный вид:

Byte 1 – 00’000’110; 06H; код операции;

Byte 2 – 00000101; 05H; операнд.

В других процессорах для пересылки непосредственного операнда используется обозначение MOV … , #05H.

LDA – (LoaD Accumulator) загрузка аккумулятора 8-разрядным операндом из ячейки памяти, адрес которой указан во втором и третьем байтах команды. Режим адресации – прямой. Команда выполняется за четыре машинных цикла и имеет код $0011’1010_2$ или 3AH. Например, команда LDA 8203H; A ← M(8203H) примет вид

Byte 1 – 0011’1010; 3AH; код операции;

Byte 2 – 0000’0011; 03H; младшая часть адреса;

Byte 3 – 1000’0010; 82H; старшая часть адреса.

STA – сохранение содержимого аккумулятора в ячейке памяти, адрес которой указан во втором и третьем байте команды. Режим адресации – прямой. Команда выполняется за четыре машинных цикла и имеет код $0011’0010_2$, или 32H.

LXI – загрузка регистровой пары 16-разрядным непосредственным операндом, содержащимся во втором и третьем байтах самой команды. Режим адресации – непосредственный. Команда выполняется за три машинных цикла и имеет формат

00’RR’0001,

где RR – обозначает код регистровой пары (см. табл. 1). Например, команда LXI H,8203H; HL ← 8203H будет иметь вид

Byte 1 – 0011’1010; 21H; код операции;

Byte 2 – 0000’0011; 03H; младшая часть 16-разрядного операнда;

Byte 3 – 1000’0010; 82H; старшая часть 16-разрядного операнда.

3.5. Команды ввода/вывода

Команды ввода/вывода позволяют переместить содержимое аккумулятора в порт вывода и, наоборот, из порта ввести данные в аккумулятор. Напомним, что микропроцессор КР580ВМ80А способен адресовать 256 портов ввода и 256 портов вывода, поэтому адреса портов могут принимать значения от 00H до FFH.

IN – ввод из порта в аккумулятор. Команда состоит из двух байт – кода операции и номера порта. Например, IN 02H; A ← Port 02H. Режим адресации – прямой.

OUT – вывод из аккумулятора в порт. Например, OUT 04H; Port 04H ← A.

3.6. Арифметические команды

Арифметические команды выполняются над содержимым аккумулятора и операндом, указанным в команде. Результат помещается в аккумулятор. КР580ВМ80А имеет следующие команды сложения/вычитания:

ADD – сложение содержимого регистра или ячейки памяти с содержимым аккумулятора;

ADC – сложение с учетом переноса;

ADI – сложение содержимого аккумулятора с непосредственным операндом;

ACI – сложение с непосредственным операндом с учетом флага С.

Например:

ADD B; A ← A + B

ADC C; A ← A + C + flag C

ADI 05H; A ← A + 05H

ACI 05H; A ← A + 05H + flag C

Сложение чисел с разрядностью более одного байта производится программными методами. Например, сложение двух 16-разрядных чисел выполняется в два этапа – сначала складываются младшие байты командой ADD, а затем старшие командой ADC (с учетом переноса).

Рассмотрим операции вычитания. В силу внутренних особенностей АЛУ не обладает возможностями вычитания, оно осуществляет сложение, представляя вычитаемое в форме дополнительного кода и затем складывая его. При арифметических операциях байт может интерпретироваться как:

1) двоичное число без знака в диапазоне от 0 до 255_{10}, или 2^8;

2) число со знаком от -128_{10} до $+127_{10}$ в котором старший (седьмой) бит означает положительность или отрицательность числа;

3) двоично-десятичное число без знака от 00 до 99;

4) двоично-десятичное число со знаком от –50 до 49.

Таким образом, весь диапазон чисел можно представить в виде круга, правую половину которого составляют положительные числа, а начиная с 80H по FFH – отрицательные (рис. 14). При кодировании отрицательных чисел используется *дополнительный код*, который может быть образован двумя способами:

1. $ДК(a) = \overline{a} + 1$, где $\overline{a}$, – это число, представленное в двоичном коде, в котором единицы инвертированы в нули и наоборот. Например, число 05H=0000’0101. Тогда –05H = = 1111’1010 + 1 = 1111’1011 = FBH.

2. $ДК(a) = FF - a + 1$. Число FFH в этой формуле – это максимальное шестнадцатеричное число, которое может быть представлено с помощью восьми бит.

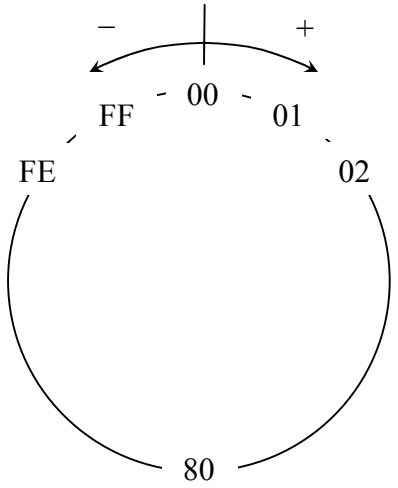

Рис. 14. Диапазон чисел

Каждая команда вычитает содержимое регистра или ячейки памяти из содержимого аккумулятора:

SUB – вычитание содержимого регистра или ячейки памяти из содержимого аккумулятора;

SBB – вычитание содержимого регистра или ячейки памяти с учетом заёма;

SUI – вычитание непосредственного операнда из содержимого аккумулятора;

SBI – вычитание непосредственного операнда из содержимого аккумулятора с заёмом.

Например:

SUB B; A ← A – B

SBB C; A ← A – C – flag C

SUI 05H; A ← A – 05H

SBI 05H; A ← A – 05H – flag C

Отдельную подгруппу арифметических команд составляют команды увеличения/уменьшения на единицу (инкремента/декремента):

INR – инкремент содержимого регистра или ячейки памяти;

DCR – декремент содержимого регистра или ячейки памяти,

например, DCR M; M(HL) ← M(HL) – 1;

INX – инкремент содержимого регистровой пары;

DCX – декремент содержимого регистровой пары.

Особенность этих команд, в отличие от команд сложения/вычитания в том, что команды INR и DCR не формируют флаг C, а команды INX и DCX вообще не формируют флаги.

При операциях ввода/вывода может потребоваться работа в десятичной системе счисления. Отличие двоично-десятичного кода от шестнадцатеричного в том, что каждая десятичная цифра кодируется одной тетрадой. Тогда $25_{10} = 0010'0101_{2/10} = 0001'1001_2 = 19H$. Микропроцессор имеет команду для преобразования числа из шестнадцатеричной системы счисления в двоично-десятичную:

DAA – десятичная коррекция аккумулятора.

Особенность этой команды в том, что она применяется только после команды шестнадцатеричного сложения ADD. Коррекция выполняется над каждой тетрадой числа следующим образом:

1) если после сложения значение младшей тетрады больше 9 или установился флаг AC, то к содержимому аккумулятора прибавляется 6;

2) если после сложения значение старшей тетрады больше 9 или установился флаг C, то к старшей тетраде прибавляется 6.

Например:

$$\begin{array}{rr} \text{ADD} & 38\text{H} \\ & \underline{54\text{H}} \\ & 8\text{CH} \\ + & 6 \\ & \underline{\quad\quad} \\ & 92\text{D} \end{array}$$

При вычитании команда DAA работает некорректно, поэтому необходимо использовать сложение с дополнительным кодом и коррекцию.

3.7. Команды логических операций

Логические команды составляют еще одну группу команд процессора КР580. Они сведены в табл. 2 и содержат команды И (логическое умножение), ИЛИ (логическое сложение), Исключающее ИЛИ, НЕ (инверсия) и сдвига. Здесь именно аккумулятор составляет ядро большинства операций. Как и в рассмотренных ранее командах, режим адресации и здесь влияет на способ и место нахождения других данных в системе.

Таблица 2

Таблица истинности логических операций

Вход	Логическое И	Логическое ИЛИ	Исключающее ИЛИ
0 0	0	0	0
0 1	0	1	1
1 0	0	1	1
1 1	1	1	0

ANA – поразрядное логическое И содержимого регистра и аккумулятора;

ANI – поразрядное логическое И содержимого аккумулятора и непосредственного операнда;

ORA – поразрядное логическое ИЛИ содержимого регистра и аккумулятора;

ORI – поразрядное логическое ИЛИ содержимого регистра и непосредственного операнда;

XRA – поразрядное логическое Исключающее ИЛИ содержимого регистра и аккумулятора (сумма по модулю два);

XRI – поразрядное логическое Исключающее ИЛИ содержимого регистра и непосредственного операнда (сумма по модулю два).

Логические команды позволяют установить в единицу, сбросить в ноль, инвертировать и проверить интересующие нас биты. **Установка бита** в единицу производится формированием *маски*, которая определяет позицию требуемого бита. Маска для установки бита – это байт с нулями во всех битах, кроме искомого. Затем выполняется операция ИЛИ с содержимым аккумулятора и полученной маской. Аналогично, **сброс** бита в ноль осуществляется операцией И над маской, с единицами во всех битах, кроме требуемого, обозначенного нулем. После этих операций следует команда перехода по условию состояния флага Z.

Для примера запишем программы изменения содержимого пятого бита регистра B:

```
;Установка в "1"      ;Сброс в "0"          ;Инверсия бита
MVI  A,00100000B      MVI  A,11011111B      MVI  A,00100000B
ORA  B                ANA  B                XRA  B
MOV  B,A              MOV  B,A              MOV  B,A
```

Другой пример – проанализируем содержимое пятого бита регистра B:

```
MOV  A,B                  b7  b6  b5  b4  b3  b2  b1  b0
ANI  00100000B            0   0   1   0   0   0   0   0
                          ---------------------------------
                          0   0   b5  0   0   0   0   0
```

Условный переход, если флаг Z = 1.

При выполнении логических команд формируются флаги S, Z, P. Флаги C, AC обнуляются. Это дает возможность определить некоторые свойства операнда в аккумуляторе, не изменяя его самого, – знак числа, равно ли оно нулю и четность/нечетность количества единиц (команды ANA A, ORA A). Команда XRA A производит обнуление аккумулятора.

CMA – инвертирование содержимого аккумулятора.

Эта команда может использоваться для вычисления дополнительного кода:

```
CMA       ; инвертирование аккумулятора;
INR  A    ; увеличение аккумулятора на единицу.
```

3.8. Команды сдвига

Микропроцессор КР580ВМ80А имеет четыре команды циклического сдвига. Арифметический и логический сдвиги формируются программно (рис. 15).

RLC – циклический сдвиг содержимого аккумулятора на одну позицию влево. Младший бит и флаг С принимают значение вытесненного бита, т.е. бывшего старшего бита;

RRC – циклический сдвиг содержимого аккумулятора на одну позицию вправо. Старший бит и флаг С принимают значение вытесненного бита, т.е. бывшего младшего бита;

RAL – циклический сдвиг содержимого аккумулятора на одну позицию влево вместе с флагом С. В младшем бите устанавливается содержимое флага С;

RAR – циклический сдвиг содержимого аккумулятора на одну позицию вправо вместе с флагом С. В старшем бите устанавливается содержимое флага С.

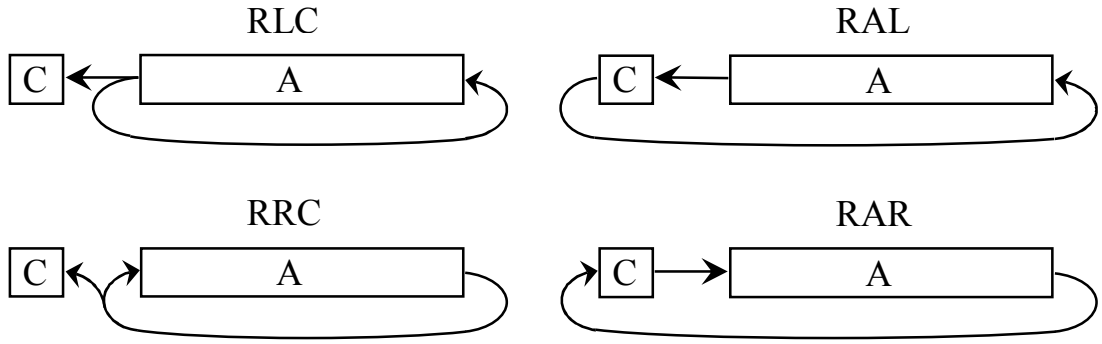

Рис. 15. Выполнение команд циклического сдвига

Эти команды позволяют реализовать арифметический сдвиг, т.е. умножение и деление на два (рис. 16).

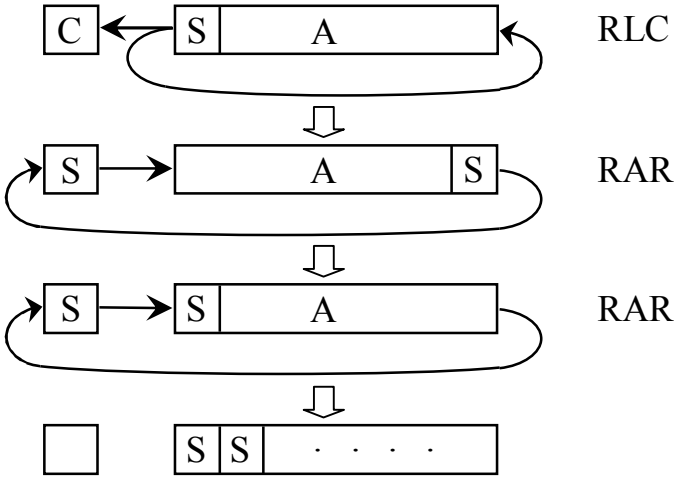

Рис. 16. Арифметический сдвиг вправо, реализующий деление на 2

Особенность арифметического сдвига состоит в неизменности старшего (знакового) разряда. Для этого необходимо выполнить циклический сдвиг влево, минуя флаг переноса, и двукратный циклический сдвиг вправо через флаг

переноса. После второго сдвига в аккумуляторе восстанавливается исходное число, а во флаге переноса помещается его знак. Третий сдвиг формирует число, соответствующее арифметическому сдвигу вправо.

Замечание. При арифметическом сдвиге вправо положительных чисел предельное значение составляет 00H; при сдвиге влево отрицательных чисел предельное значение составляет FFH = –1.

3.9. Команды сравнения

Команды сравнения используются для формирования флагов перед условным переходом. При этом выполняется вычитание, но его *результат не запоминается в аккумуляторе*.

CMP – сравнение содержимого регистра или ячейки памяти и аккумулятора;

CPI – сравнение байта с содержимым аккумулятора,

например:

CMP B; flags (A – B);

CPI 05H; flags (A – 05H).

3.10. Команды передачи управления

Эта группа содержит команды перехода по какому-либо условию, изменяя последовательный ход программы. Имеются команды двух типов: условного и безусловного переходов. Безусловный переход просто выполняют операцию, определенную счетчиком команд; условные – проверяют состояние одного из флагов процессора для определения необходимости в ветвлении.

JMP – безусловный переход. Управление передается команде по адресу, указанному во втором и третьем байтах команды;

JZ – переход, если флаг Z = 1;

JNZ – переход, если флаг Z = 0;

JC – переход, если флаг C = 1;

JNC – переход, если флаг C = 0;

JP – переход, если флаг S = 0;

JM – переход, если флаг S = 1;

JPE – переход по четности, если флаг P = 1;

JPO – переход по нечетности, если флаг P = 0,

например:

JMP 8200H; (PC) ← (байт 3) (байт 2)

Примечание. В других процессорах имеются переходы с коротким адресом, а также условные переходы по комбинациям флагов (с учетом переполнения).

3.11. Команды работы с подпрограммами

Как и команды передачи управления, эта группа команд позволяет осуществить безусловный переход либо вызов подпрограммы по условию. Иногда их называют командами переходов с возвратом.

CALL – безусловный вызов подпрограммы.

По команде CALL текущее значение счетчика команд записываются в стек, а в счетчик команд загружается новый адрес (из второго и третьего байт команды).

RET – возврат из подпрограммы. Содержимое вершины стека переписывается в счетчик команд.

Процессор КР580ВМ80А имеет также команды вызова подпрограмм и возврата из них по некоторому условию (состоянию флагов), которые аналогичны командам перехода. Например, команда CNZ осуществляет вызов подпрограммы при условии $Z = 0$, а команда RC производит возврат из подпрограммы при условии $C = 1$.

3.12.Специальные команды

DI – запрет прерываний;

EI – разрешение прерываний;

HLT – останов (до появления прерывания);

NOP – пустая операция.

Примечание. Последовательность команд DI … HLT приводит к зависанию, выход из которого возможен по сигналу «Сброс».